This paper is the draft of the following published paper:
M. Inuiguchi and T. Tanino: On Computation Methods for a Minimax Regret Solution Based on Outer Approximation and Cutting Hyperplanes, International Journal of Fuzzy Systems, Vol.3, No.4, pp.548-557 (2001)

On Computation Methods for a Minimax Regret Solution Based on Outer Approximation and Cutting Hyperplanes

Masahiro Inuiguchi and Tetsuzo Tanino

Abstract

In this paper, we discuss computation methods for minimax regret solutions to linear programming problems whose objective coefficient vectors are not known exactly but guaranteed to lie in polytopes. A solution algorithm for a minimax regret problem has been proposed based on the relaxation procedure. However, in the algorithm, we should solve non-concave sub-problems sequentially. To the non-concave sub-problem, many approaches including two phase and two-level programming approaches have been proposed. As new approaches, we discuss applications of an outer approximation method and a cutting plane method to the sub-problem. Moreover, a combination of the outer approximation and cutting hyperplane methods is proposed. We compare the computational efficiency of the solution algorithms by a numerical experiment. The results show that the outer approximation method and its combination with the cutting hyperplane method are the most efficient.

Keywords: ***Minimax Regret Solution, Linear Programming, Uncertainty, Outer Approximation, Cutting Hyperplane***

1.Introduction

In this paper, we discuss computation methods for minimax regret solutions to a linear programming problem whose objective function coefficient vector is not known exactly but in a single polytope. To such a problem, a possibly optimal solution and a necessarily optimal solution are proposed as natural extensions of an optimal solution to the conventional programming problem (see Inuiguchi and Sakawa [1]). However, a lot of possibly optimal solutions usually exist and, in many cases, no necessarily optimal solution exists.

The minimax regret solution has been proposed as a possibly optimal solution which minimizes the deviation from the necessary optimality (see Inuiguchi and Sakawa [2, 3]). It is a necessarily optimal solution when the latter exists. Because of those good properties, a minimax regret solution is regarded as a reasonable solution to a programming problem with uncertain objective function coefficients (see Inuiguchi and Sakawa [2, 3]).

However, computation methods of the minimax regret solution have not yet investigated considerably. A few computation methods were proposed for linear programming problems with interval objective function coefficients (see Inuiguchi and Sakawa [2] and Mausser and Laguna [4]) and extended to the case where the possible range of the objective function coefficient vector is given as a polytope (see Inuiguchi and Sakawa [3]). All methods adopt the same relaxation procedure including the maximum regret problem as a sub-problem. The difference among the previous computation methods of the minimax regret solution is in the solution method for the sub-problem. The sub-problem is a bilinear programming problem and at the same time a convex maximization problem. Thus, many approaches may be conceivable.

The aim of this paper is twofold: (i) to apply an outer approximation method and a cutting hyperplane method to the sub-problem and (ii) to compare the computation methods for minimax regret solutions by a numerical experiment. We restrict ourselves to the discussion among the solution methods which are effective even when the possible range of the objective function coefficient vector is given as a polytope. Thus a method proposed by Mausser and Laguna [4] is not discussed in this paper since it is applicable only when the polytope is a box-set. An outer approximation algorithm and a cutting hyperplane algorithm are specifically designed to solve the sub-problem. Moreover, a combination of the outer approximation and cutting hyperplane methods is proposed. We intended to compare the five methods by the computation time (CPU time). However, in the preliminary experiment, the cutting hyperplane method was not comparative at all because it took a lot of computation time for the convergence to an optimal

Corresponding Author: Masahiro Inuiguchi is with Department of Electronics and Information Systems, Graduate School of Engineering, Osaka University, 2-1 Yamadaoka, Suita, Osaka 565-0871, Japan, Tel: +81-6-6879-7787, Fax: +81-6-6879-7939
Email: inuiguti@eie.eng.osaka-u.ac.jp

solution to the subproblem which was solved multiple times. Then a simple modification of the relaxation procedure is proposed. In this modification, we do not calculate an optimal solution to the subproblem but test the optimality of the temporal solution for the minimax regret problem. By this modification, it is sure to reduce the computation time for solving the subproblem but is possible to increase the number of iterations in the main routine of the whole solution algorithm. Therefore there is no guarantee in the reduction of the computation time of minimax regret solution. Moreover this modification will be effective only for the methods which take a lot of computation time for solving the subproblem. In this sense, the computation time of two-phase approach will not be improved by this modification since the subproblem can be solved easily in the approach.

As the result, four methods other than the cutting hyperplane method without modification and four methods other than the two-phase approach with modification are compared. It is shown that the outer approximation approach with modification and the combination of the outer approximation and cutting hyperplane methods with modification are most computationally efficient.

2.Minimax Regret Solution

2.1 Problem statement

Let us consider the following linear programming problem with uncertain objective function coefficients:

$$\begin{aligned} &\text{maximize} && \boldsymbol{\gamma}^{\mathrm{T}}\boldsymbol{x} \\ &\text{subject to} && \boldsymbol{x}\in X=\{\boldsymbol{x}\mid A\boldsymbol{x}\le\boldsymbol{b},\boldsymbol{x}\ge 0\}, \end{aligned} \tag{1}$$

where A is an $m \times n$ matrix, $\boldsymbol{x} = (x_1,x_2,...,x_n)^{\mathrm{T}}$, $\boldsymbol{\gamma}=(\gamma_1,\gamma_2,...,\gamma_n)^{\mathrm{T}}$ and $\boldsymbol{b} = (b_1,b_2,...,b_m)^{\mathrm{T}}$. The feasible region X is assumed to be bounded.**x** is the decision variable vector. $\boldsymbol{\gamma}$ is a possibilistic variable vector and its range Γ is given as

$$\Gamma=\{\boldsymbol{c}=(c_1,c_2,...,c_n)^{\mathrm{T}}\mid D\boldsymbol{c}\le\boldsymbol{g}\}, \tag{2}$$

where D is a pxn matrix and $\boldsymbol{g}=(g_1,g_2,...,g_p)^{\mathrm{T}}$. Γ is assumed to be bounded. Thus, Γ is a polytype. Let $S(\boldsymbol{c})=\{\boldsymbol{y}\in X\mid \boldsymbol{c}^{\mathrm{T}}\boldsymbol{y}=\max_{x\in X}\boldsymbol{c}^{\mathrm{T}}\boldsymbol{x}\}$, i.e., the optimal solution set of a linear programming problem with an objective function coefficient vector $\boldsymbol{c}$. Using $S(\boldsymbol{c})$, we can define the following two kinds of optimal solution sets (see Inuiguchi and Sakawa [1]):

$$NS=\bigcap_{\boldsymbol{c}\in\Gamma}S(\boldsymbol{c}) \tag{3}$$

$$\Pi S=\bigcup_{\boldsymbol{c}\in\Gamma}S(\boldsymbol{c}) \tag{4}$$

An element of NS is a solution of Problem (1) optimal for all $\boldsymbol{c}\in\Gamma$ and called a necessarily optimal solution. On the other hand, an element of ΠS is a solution of Problem (1) optimal for at least one $\boldsymbol{c}\in\Gamma$ and called a possibly optimal solution. In many problems, there exists no necessarily optimal solution and there exist a lot of possibly optimal solutions. Thus, both of the optimal solution sets are not always practically useful. A minimax regret solution which is a possibly optimal solution minimizes a deviation from the necessary optimality has been proposed (see Inuiguchi and Sakawa [2]). A minimax regret solution is necessarily optimal if a necessarily optimal solution exists.

2.2 Minimax regret solution

Suppose that the true objective function coefficient vector is known as **c** after a solution **x** is selected. Under this supposition, the decision maker may feel a regret defined by

$$r(\boldsymbol{x},\boldsymbol{c})=\max_{\boldsymbol{y}\in X}(\boldsymbol{c}^{\mathrm{T}}\boldsymbol{y}-\boldsymbol{c}^{\mathrm{T}}\boldsymbol{x}). \tag{5}$$

$r(\boldsymbol{x},\boldsymbol{c})$ is the difference between the objective function value $\boldsymbol{c}^{\mathrm{T}}\boldsymbol{x}$ and the optimal value under the objective function coefficient vector $\boldsymbol{c}$. At the decision stage, the objective function coefficient vector is still unknown and then the worst (maximum) regret of a solution $\boldsymbol{x}$ is obtained as

$$R(\boldsymbol{x})=\max_{\boldsymbol{c}\in\Gamma}r(\boldsymbol{x},\boldsymbol{c})=\max_{\substack{\boldsymbol{c}\in\Gamma\\ \boldsymbol{y}\in X}}(\boldsymbol{c}^{\mathrm{T}}\boldsymbol{y}-\boldsymbol{c}^{\mathrm{T}}\boldsymbol{x}). \tag{6}$$

The smaller $R(\boldsymbol{x})$ is, the better **x** is. Therefore, in view of the regret, Problem (1) is formulated as a minimax regret problem,

$$\underset{\boldsymbol{x}\in X}{\text{minimize}}\ \max_{\substack{\boldsymbol{c}\in\Gamma\\ \boldsymbol{y}\in X}}(\boldsymbol{c}^{\mathrm{T}}\boldsymbol{y}-\boldsymbol{c}^{\mathrm{T}}\boldsymbol{x}). \tag{7}$$

When $R(\boldsymbol{x}) = 0$, $\boldsymbol{x}$ is a necessarily optimal solution. Moreover, a solution of Problem (7) is a possibly optimal solution (see Inuiguchi and Kume [5]).

3. Previous Solution Methods

3.1 Relaxation Procedure

Since the minimax regret problem (7) is a minimax problem with separable constraints, we can apply a relaxation procedure [6]. Applying the relaxation procedure with an admissible error ε we obtain the following solution algorithm:

[Relaxation Procedure Algorithm]

Step 1 Select $\boldsymbol{c}^0 \in \Gamma$ arbitrarily. Obtain $\boldsymbol{z}^0 \in S(\boldsymbol{c}^0)$.

Step 2 Set $r^0 = 0,\ k = 1$ and $\boldsymbol{x}^0 = \boldsymbol{z}^0$.

Step 3 Obtain an optimal solution $(\boldsymbol{c}^k, \boldsymbol{z}^k)$ and the optimal value r^k of a maximum regret problem,

$$\underset{\substack{\boldsymbol{c}\in\Gamma \\ \boldsymbol{y}\in X}}{\text{maximize}} \left(\boldsymbol{c}^{\mathrm{T}}\boldsymbol{y} - \boldsymbol{c}^{\mathrm{T}}\boldsymbol{x}^0\right). \quad \textbf{(8)}$$

Step 4 if $r^k \le r^0 + \varepsilon$, then terminate the algorithm. The obtained minimax regret solution is $\boldsymbol{x}^0$.

Step 5 Update $(\boldsymbol{x}^0, r^0)$ by an optimal solution $(\boldsymbol{x}^*, r^*)$ of

$$\begin{aligned} &\text{minimize} \quad r \\ &\text{subject to} \\ &\left(\boldsymbol{c}^j\right)^T \boldsymbol{z}^j - \left(\boldsymbol{c}^j\right)^T \boldsymbol{x} \le r,\ j = 0,1,2,\ldots,k, \\ &A\boldsymbol{x} \le \boldsymbol{b},\ \boldsymbol{x} \ge 0. \end{aligned} \quad (9)$$

Update $k = k + 1$ and return to Step 3.

The problems at Steps 1 and 5 are linear programming problems. Thus, they can be solved easily. However, the problem at Step 3 is a non-convex programming problem. The previous solution methods for Problem (7) are different in the solution methods for the sub-problem (8). In what follows, we briefly review two solution methods for Problem (8).

3.2 Two-Phase Approach

Let ΠB be the set of all possibly optimal extreme points of Problem (1) and let

$$f(\boldsymbol{y}) = \max_{\boldsymbol{c}\in\Gamma} \left(\boldsymbol{c}^{\mathrm{T}}\boldsymbol{y} - \boldsymbol{c}^{\mathrm{T}}\boldsymbol{x}^0\right). \quad (10)$$

Then maximum regret problem (8) can be represented as

$$\underset{\boldsymbol{y}\in\Pi B}{\text{maximize}}\ f(\boldsymbol{y}). \quad (11)$$

Since X is bounded, ΠB is a finite set. All the elements of ΠB can be enumerated by the method proposed in Inuiguchi, Higashitani and Tanino [7]. On the other hand, $f(\boldsymbol{y})$ is easily obtained by solving a linear programming problem. Hence, after enumerating all elements of ΠB, Problem (8) can be solved easily by obtaining $f(\boldsymbol{y}^j)$ for all $\boldsymbol{y}^j \in \Pi B$ (see Inuiguchi and Sakawa [2, 3]).

3.3 Bilevel Programming Approach

Since Problem (8) is equivalent to $\text{maximize}_{\boldsymbol{c}\in\Gamma,\boldsymbol{y}\in\Pi S} \left(\boldsymbol{c}^T\boldsymbol{y} - \boldsymbol{c}^T\boldsymbol{x}^0\right)$, it can be seen as a bilevel programming problem,

$$\begin{aligned} &\underset{\boldsymbol{c},\,\boldsymbol{y}}{\text{maximize}} \quad \left(\boldsymbol{c}^{\mathrm{T}}\boldsymbol{y} - \boldsymbol{c}^{T}\boldsymbol{x}^{0}\right), \\ &\text{subject to} \quad D\boldsymbol{c} \le \boldsymbol{g}, A\boldsymbol{y} \le \boldsymbol{b}, \boldsymbol{y} \ge 0, \\ &\qquad \mathbf{c}^T\mathbf{y} = \max_{\mathbf{z}} \mathbf{c}^T\mathbf{z}, \\ &\qquad \text{subject to} \quad A\boldsymbol{z} \le \boldsymbol{b}, \boldsymbol{z} \ge 0. \end{aligned} \quad (12)$$

Replacing the lower level programming problem with its optimality condition and introducing slack variables $\boldsymbol{w} = \boldsymbol{b} - A\boldsymbol{y} \ge 0$, surplus variables $\boldsymbol{s} = A^{\mathrm{T}}\boldsymbol{u} - \boldsymbol{c} \ge 0$ and a redundant constraint, $\boldsymbol{b}^{\mathrm{T}}u \le \left(\boldsymbol{c}^{\mathrm{R}}\right)^{\mathrm{T}}\boldsymbol{y}$, the problem can be reduced to

$$\begin{aligned} &\underset{\boldsymbol{y},\boldsymbol{u},\boldsymbol{s}}{\text{maximize}} \left(\boldsymbol{x}^0\right)^{\mathrm{T}}\boldsymbol{s} + \left(\boldsymbol{w}^0\right)^{\mathrm{T}}\boldsymbol{u}, \\ &\text{subject to} \quad DA^{\mathrm{T}}\boldsymbol{u} - D\boldsymbol{s} \le \boldsymbol{g}, A\boldsymbol{y} + \boldsymbol{w} = \boldsymbol{b}, \\ &\qquad \boldsymbol{u} \ge 0, \boldsymbol{w} \ge 0, \boldsymbol{y} \ge 0, \boldsymbol{s} \ge 0, \\ &\qquad \boldsymbol{b}^{\mathrm{T}}\boldsymbol{u} \le \left(\boldsymbol{c}^{\mathrm{R}}\right)^{\mathrm{T}}\boldsymbol{y},\ \boldsymbol{u}^{\mathrm{T}}\boldsymbol{w} + \boldsymbol{s}^{\mathrm{T}}\boldsymbol{y} = 0, \end{aligned} \quad (13)$$

where we define $\boldsymbol{w}^0 = \boldsymbol{b} - A\boldsymbol{x}^0$ and $\boldsymbol{c}^{\mathrm{R}} = \left(c_1^{\mathrm{R}}, c_2^{\mathrm{R}}, \ldots, c_n^{\mathrm{R}}\right)^{\mathrm{T}}$ with $c_i^{\mathrm{R}} = \max_{c\in\Gamma} c_i$.

Let $\bar{\boldsymbol{s}} = \left(\bar{s}_1, \bar{s}_2, \ldots, \bar{s}_{n+m}\right)^{\mathrm{T}} = \left(\boldsymbol{s}^{\mathrm{T}}, \boldsymbol{u}^{\mathrm{T}}\right)^{\mathrm{T}}$ and $\bar{\boldsymbol{y}} = \left(\bar{y}_1, \bar{y}_2, \ldots, \bar{y}_{n+m}\right)^{\mathrm{T}} = \left(\boldsymbol{y}^{\mathrm{T}}, \boldsymbol{w}^{\mathrm{T}}\right)^{\mathrm{T}}$. The last constraint of (12) can be written as $\bar{\boldsymbol{s}}^{\mathrm{T}}\bar{\boldsymbol{y}} = 0$. Because of the non-negativity, this constraint becomes complementary constraints, $\bar{s}_j \bar{y}_j = 0$, j=1,2,…,n+m. Moreover, dropping the last constraint, $\boldsymbol{u}^{\mathrm{T}}\boldsymbol{w} + \boldsymbol{s}^{\mathrm{T}}\boldsymbol{y} = 0$, Problem (12) becomes a linear programming problem. Thus, an application of a branch and bound method is conceivable by solving the relaxed linear programming problem and introducing a constraint $\bar{s}_j = 0$ or $\bar{y}_j = 0$ until all possible combinations are exhausted (see Inuiguchi and Sakawa [3]).

4. Proposed Approaches

4.1 Outer Approximation

Using the function f defined by (10), Problem (8) can be written as

$$\underset{\boldsymbol{y}\in X}{\text{maximize}}\ f(\boldsymbol{y}). \quad (14)$$

We can prove that f is a convex function.

Theorem 1 *f is a convex function.*

Proof Let $\boldsymbol{y}^1, \boldsymbol{y}^2 \in X$. For any $\lambda \in [0,1]$, we have

$$
\begin{aligned}
&f\left(\lambda \boldsymbol{y}_1+(1-\lambda)\boldsymbol{y}_2\right)\\
&=\max_{\boldsymbol{c}\in\Gamma}\left(\lambda \boldsymbol{c}^{\mathrm{T}}\boldsymbol{y}_1+(1-\lambda)\boldsymbol{c}^{\mathrm{T}}\boldsymbol{y}_2-\boldsymbol{c}^{\mathrm{T}}\boldsymbol{x}^0\right)\\
&\le \lambda\max_{\boldsymbol{c}\in\Gamma}\left(\boldsymbol{c}^{\mathrm{T}}\boldsymbol{y}_1-\boldsymbol{c}^{\mathrm{T}}\boldsymbol{x}^0\right)+(1-\lambda)\max_{\boldsymbol{c}\in\Gamma}\left(\boldsymbol{c}^{\mathrm{T}}\boldsymbol{y}_2-\boldsymbol{c}^{\mathrm{T}}\boldsymbol{x}^0\right)\\
&=\lambda f\left(\boldsymbol{y}_1\right)+(1-\lambda)f\left(\boldsymbol{y}_2\right).
\end{aligned}
$$

Thus, f is a convex function.

Q.E.D.

From Theorem 1, we know that Problem (14) is a convex maximization problem and thus, an optimal solution exists in the set of extreme points of X (see, for example, Horst and Tuy [8]). From this point of view, we can apply any solution algorithm for convex maximization problems. In this paper, we apply an outer approximation method because the fact that X is a polytope guarantees the termination in a finite number of iterations.

Taking advantage of the fact that the optimal solution is in $\prod B$, the outer approximation algorithm for Problem (14) is obtained as follows.

[Outer Approximation Algorithm]

Step 1 Initialize p=0 and obtain a polytope Y_0 such that $X \subseteq Y_0$.

Step 2 Enumerate all the elements of $\Pi B(Y_p)$.

Step 3 Calculate $f(\boldsymbol{y})$ for all $\boldsymbol{y} \in \Pi B(Y_p)$. Let $\boldsymbol{y}^p$ be a solution which maximizes $f(\boldsymbol{y})$ subject to $\boldsymbol{y} \in \Pi B(Y_p)$. Moreover, let $\boldsymbol{d}^p$ be a $\boldsymbol{c} \in \Gamma$ such that $f\left(\boldsymbol{y}^p\right)=\boldsymbol{c}^T\left(\boldsymbol{y}^p-\boldsymbol{x}^0\right)$.

Step 4 If $f\left(\boldsymbol{y}^p\right) \le r^0$, terminate the algorithm with setting $r^k = r^0$.

Step 5 If $\boldsymbol{y}^p \in X$, terminate the algorithm with setting $\boldsymbol{c}^k = \boldsymbol{d}^p$, $\boldsymbol{z}^k = \boldsymbol{y}^p$ and $r^k = f\left(\boldsymbol{y}^p\right)$.

Step 6 Solve a linear programming problem,

$$\underset{\boldsymbol{y}\in X}{\text{maximize}}\ \boldsymbol{d}^{p\mathrm{T}}\boldsymbol{y}, \tag{15}$$

and let $\boldsymbol{w}^p$ be an optimal solution. Let Z be a set defined by constraints whose corresponding slack variables are nonbasic at the optimal solution $\boldsymbol{w}^p$.

Step 7 Update $Y_{p+1} = Y_p \cap Z$ and $p = p+1$. Return to Step 2.

In the algorithm above, $\Pi B(Y_p)$ is the set of all possibly optimal extreme points of Problem (1) when X is replaced with Y_p.

Let us discuss the initialization of Y_p in Step 1. Since the above outer approximation algorithm is proposed for solving the sub-problem (8), it is called many times. Once the algorithm is called, Y_p always satisfies $Y_p \supseteq X$. Thus, the latest Y_p of the previous call of the outer approximation algorithm can be utilized as the initialization of Y_0 at the current call. Now, we discuss how we initialize Y_0 at the first call of the outer approximation algorithm.

Before the first call of the outer approximation algorithm, we calculate an optimal solution $\boldsymbol{z}^0$ of a linear programming problem at Step 1 of the relaxation procedure algorithm (see Subsection 3.1). Utilizing this solution, we define Y_0 by all constraints whose slack variables are nonbasic at $\boldsymbol{z}^0$. However, there is no guarantee that Y_0 is bounded. To ensure the boundedness, we add the following constraint,

$$\boldsymbol{e}^{\mathrm{T}}\boldsymbol{x} \le \max_{\boldsymbol{y}\in X}\boldsymbol{e}^{\mathrm{T}}\boldsymbol{y}, \tag{16}$$

where $\boldsymbol{e}=(1,1,\ldots,1)^{\mathrm{T}}$. The right-hand side problem can be solved easily by a post-optimality technique of the simplex method from the basis associated with $\boldsymbol{z}^0$.

Now let us discuss the enumeration of all elements of $\Pi B(Y_p)$ at Step 2. $\Pi B(Y_0)$ at the first call can be obtained by the algorithm proposed by Inuiguchi, Higashitani and Tanino [7]. $\Pi B(Y_0)$ after the second call has been already obtained at the last call. Hence, we discuss the method for updating $\Pi B(Y_p)$ at Step 2 for $p \ge 2$. Utilizing the basic solution $\boldsymbol{w}^{p-1}$ obtained at Step 6, this can be done by the following algorithm.

[Algorithm for Updating $\Pi B(Y_p)$]

Step 1 Erase all elements which do not satisfy the additional constraints from $\Pi B(Y_{p-1})$.

Step 2 From the basic solution $\boldsymbol{w}^{p-1}$ at Step 6, obtain a basic expression of $\boldsymbol{w}^{p-1}$ with respect to Y_p. Enumerate all possibly optimal extreme points of Y_p as far as at least one slack variable corresponding to one of additional constraints is nonbasic. Let Δ be the set of enumerated extreme points. Let $\Pi B(Y_p) = \Pi B(Y_{p-1}) \cup \Delta$.

4.2 Cutting Hyperplanes

The sub-problem (8) is a bilinear programming problem, thus we apply the cutting hyperplane method developed by Sherali and Shetty [9] which guarantees the convergence in a finite number of iterations. In this method, two kinds of cutting hyperplanes are considered and a cutting hyperplane is introduced at each iteration.

We denote the *i*-th cut by $\boldsymbol{t}^{i\mathrm{T}}\boldsymbol{x} \le t_0^i$. Let *Q* be a feasible region of *q* cuts composed of all previously obtained cuts, i.e., $Q = \left\{\boldsymbol{x} \mid \boldsymbol{t}^{i\mathrm{T}}\boldsymbol{x} \le t_0^i, i = 1,2,..., q\right\}$. First, an extreme face of *X* with respect to *Q* is calculated by repetitive use of the simplex method. If the obtained extreme face is not an extreme point of *X*, we introduce a disjunctive face cut. Otherwise, we introduce a negative-edge extension polar cut.

If the obtained extreme face is not an extreme point of *X*, a basic solution $y^0 = \left(y_1^0, y_2^0, ..., y_n^0\right)^{\mathrm{T}}$ on the extreme face is obtained at the same time through the extreme face determination algorithm. Let $B_N = \left\{r \mid y_r^0 > 0\right\}$ and *J* be a set of indices of nonbasic variables with respect to y^0. Choosing $r \in B_N$ arbitrarily, we solve $\text{minimize}_y \left\{y_r \mid y \in X \cap Q\right\}$ by a simplex method with a restriction that slack variables $s_i = t_0^i - \boldsymbol{t}^{i\mathrm{T}}\boldsymbol{y} \ge 0, i = 1,2,..., q$ are eligible to enter the basis. Then, we obtain the following canonical form of a variable y_r;

$$y_r + \sum_{j \in J} \alpha_{rj} y_j + \sum_{i \in S_r} \beta_{ri} s_i = \gamma_r, \tag{17}$$

where S_r be the set of indices of all nonbasic slack variables, i.e., s_i's. Note that *J* includes all indices of nonbasic variables y_j's at the optimal solution and some α_{rj}'s are zeros.

Repeating the above procedure, we obtain canonical forms (17) for every $r \in B_N$. Let B_N^* be a subset of B_N obtained by deleting all $r \in B_N$ such that $\alpha_{rj} < 0$ for all $j \in J$. We define $S_N^* = \bigcup_{r \in B_N^*} S_r$ and

$$\delta_r = \min\left(\min_{j \in J: \alpha_{rj} > 0}\left(\frac{\gamma_r}{\alpha_{rj}} \max_{r \in B_N^*} \frac{\alpha_{rj}}{\gamma_r}\right), \min_{i \in S_N^*: \beta_{ri} > 0}\left(\frac{\gamma_r}{\beta_{ri}} \max_{r \in B_N^*} \frac{\beta_{ri}}{\gamma_r}\right)\right), \tag{18}$$

where we define $\beta_{ri} = 0, i \in S_N^* \setminus S_r$. Then a disjunctive face cut is obtained as

$$\sum_{j \in J}\left(\max_{r \in B_N^*} \delta_r \frac{\alpha_{rj}}{\gamma_r}\right) y_j + \sum_{i \in S_N^*}\left(\max_{r \in B_N^*} \delta_r \frac{\beta_{ri}}{\gamma_r}\right) s_i \ge 1. \tag{19}$$

If the obtained extreme face is an extreme point of *X*, we calculate a weak pseudo-global minimum $(\hat{\boldsymbol{c}}, \hat{\boldsymbol{y}})$ such that $\hat{\boldsymbol{y}}$ is a basic solution with respect to *X*, $\hat{\boldsymbol{c}}^{\mathrm{T}}\left(\hat{\boldsymbol{y}} - \boldsymbol{x}^0\right) = \max_{\boldsymbol{c} \in \Gamma} \boldsymbol{c}^{\mathrm{T}}\left(\hat{\boldsymbol{y}} - \boldsymbol{x}^0\right)$ and $\max_{\boldsymbol{c} \in \Gamma} \boldsymbol{c}^{\mathrm{T}}\left(\hat{\boldsymbol{y}} - \boldsymbol{x}^0\right) \le \hat{\boldsymbol{c}}^{\mathrm{T}}\left(\hat{\boldsymbol{y}} - \boldsymbol{x}^0\right)$ for all basic solutions $\bar{\boldsymbol{y}}$ adjacent to $\hat{\boldsymbol{y}}$. Then the following two parameters $\hat{\lambda}_j$ and $\tilde{\lambda}_j$ are computed by a modified Newton method.

$$\hat{\lambda}_j = \sup\left\{\lambda \mid \max_{\boldsymbol{c} \in \Gamma} \boldsymbol{c}^{\mathrm{T}}\left(\hat{\boldsymbol{y}} - \boldsymbol{e}^j \lambda - \boldsymbol{x}^0\right) \le r^k\right\}, \tag{20}$$

$$\tilde{\lambda}_j = \sup\left\{\lambda \mid \max_{\boldsymbol{c} \in \Gamma} \boldsymbol{c}^{\mathrm{T}}\left(\hat{\boldsymbol{y}} + \boldsymbol{e}^j \lambda - \boldsymbol{x}^0\right) \le r^k\right\}, \tag{21}$$

where r^k is the current maximum regret and $\boldsymbol{e}^j$ is the extended column of the nonbasic variable x_j. The *i*-th component of $\boldsymbol{e}^j$ shows the negative rate of change of *i*-th component of $\boldsymbol{y}$ with respect to a unit increment of nonbasic variable x_j. Let

$$\bar{\lambda}_j = \begin{cases} \hat{\lambda}_j & \text{if} \quad \hat{\lambda}_j \ne +\infty \\ \tilde{\lambda}_j & \text{if} \quad \hat{\lambda}_j = +\infty \end{cases} \tag{22}$$

Then the negative-edge extension polar cut is obtained as

$$\sum_{j \in J} \frac{y_j}{\bar{\lambda}_j} \ge 1, \tag{23}$$

where *J* is an index set of nonbasic variables with respect to $\hat{\boldsymbol{y}}$.

The cutting hyperplane algorithm becomes as follows.

[Cutting Hyperplane Algorithm]

Step 1 Set $Q = \mathbf{R}^n$ and $r^k = r^0$.

Step 2 Obtain an extreme face of *X* with respect to *Q*. If it does not exist, the current solution is optimal and terminate the algorithm. If the extreme face is an extreme point of *X*, go to Step 4.

Step 3 Calculate B_N^*. If $B_N^* = \phi$, the current solution is optimal and terminate the algorithm. Otherwise, obtain a disjunctive face cut (19) and update *Q* by introducing this cutting hyperplane. Return to Step 2.

Step 4 Obtain a weak pseudo-global minimum $(\hat{\boldsymbol{c}}, \hat{\boldsymbol{y}})$. If $\hat{\boldsymbol{c}}^{\mathrm{T}}\left(\hat{\boldsymbol{y}} - \boldsymbol{x}^0\right) > r^k$ then reset $\left(\boldsymbol{c}^k, \boldsymbol{y}^k\right) = (\hat{\boldsymbol{c}}, \hat{\boldsymbol{y}})$ and $r^k = \hat{\boldsymbol{c}}^{\mathrm{T}}\left(\hat{\boldsymbol{y}} - \boldsymbol{x}^0\right)$. Let *J*

be the index set of nonbasic variables at $\hat{\boldsymbol{y}}$ with respect to X. For every $j \in J$, obtain $\hat{\lambda}_j$ and if it is $+\infty$, obtain $\tilde{\lambda}_j$. If there exists $\hat{\lambda}_j = 0,\ j \in J$, Return to Step 3. If $\hat{\lambda}_j = +\infty$, for all $j \in J$ the current solution is optimal and terminate the algorithm. Otherwise, establish a negative-edge extension polar cut (23) and update Q by introducing this cutting hyperplane. Return to Step 2.

4.3 Combination of Outer Approximation and Cutting Hyperplanes

In the cutting hyperplane method, computational efficiency is enhanced by erasing the region in which no optimal solution exists. However, there is no upper bound information about the optimal value, so that a lot of computation may be required before the termination criterion is satisfied. On the other hand, in the outer approximation method, upper bound information is utilized but there is no cut down of feasible region. Hence unnecessary extreme points may be enumerated. Because of the complementary properties of the cutting hyperplane method and the outer approximation method, an approach as a combination of those methods is conceivable. Then we obtain the following algorithm.

[Algorithm by the Combination of Outer Approximation and Cutting Hyperplanes]

Step 1 Initialize p=0 and obtain a polytope Y_0 such that $X \subseteq Y_0$. Set $Z = \mathbf{R}^n$ and $r^k = r^0$.

Step 2 Enumerate all elements of $\Pi B(Y_p)$.

Step 3 Calculate $f(\boldsymbol{y})$ for all $\boldsymbol{y} \in \Pi B(Y_p)$. Let $\boldsymbol{y}^p$ be an element $\boldsymbol{y}$ of Y_p which maximizes $f(\boldsymbol{y})$. Moreover, let $\boldsymbol{d}^p$ be a $\boldsymbol{c} \in \Gamma$ such that $f(\boldsymbol{y}^p) = \boldsymbol{c}^{\mathrm{T}}(\boldsymbol{y}^p - \boldsymbol{x}^0)$.

Step 4 If $f(\boldsymbol{y}^p) \le r^k$, terminate the algorithm.

Step 5 If $\boldsymbol{y}^p \in X$, terminate the algorithm with setting $\boldsymbol{c}^k = \boldsymbol{d}^p, \boldsymbol{z}^k = \boldsymbol{y}^p$ and $r^k = f(\boldsymbol{y}^p)$.

Step 6 Solve a linear programming problem,

$$\underset{\boldsymbol{y} \in Y_p \cap X}{\text{maximize}}\ \boldsymbol{d}^{p\mathrm{T}} \boldsymbol{y}, \tag{24}$$

and let $\boldsymbol{w}^p$ be an optimal solution. If $\boldsymbol{w}^p$ is not an extreme point of X, go to Step 10.

Step 7 Starting from $\boldsymbol{w}^p$, obtain a weak pseudo-global minimum $(\hat{\boldsymbol{c}}, \hat{\boldsymbol{y}})$.

Step 8 Let J be the index set of nonbasic variables at $\hat{\boldsymbol{y}}$ with respect to X. For every $j \in J$, obtain $\hat{\lambda}_j$ and if it is $+\infty$, obtain $\tilde{\lambda}_j$. If there exists $\hat{\lambda}_j = 0,\ j \in J$, go to Step10. If $\hat{\lambda}_j = +\infty$, for all $j \in J$ the current solution is optimal and terminate the algorithm. Otherwise, establish a negative-edge extension polar cut (23) and update Z by introducing this cutting hyperplane.

Step 9 If $\hat{\boldsymbol{y}} \notin Z$, go to Step 11.

Step 10 Let Z' be a set of solutions satisfying constraints whose corresponding slack variables are nonbasic at the optimal solution $\mathbf{w}^p$. Reset $Z = Z \cap Z'$.

Step 11 Set $Y_{p+1} = Y_p \cap Z$. Return to Step 2.

The details of the steps are mostly the same as those in outer approximation algorithm and cutting hyperplane algorithm. A big difference is in Step 2 returned from Step 11. The difference is as follows: in the outer approximation algorithm, the starting point of the enumeration has already obtained as $\boldsymbol{w}^{p-1}$ of the previous iteration and $\Pi B(Y_p)$ can be updated from $\Pi B(Y_{p-1})$ by the algorithm described in Subsection 4.1. On the other hand, in the algorithm of the combination, because there is no guarantee of $\boldsymbol{w}^{p-1} \in Y_p$, $\boldsymbol{w}^{p-1}$ cannot be used as a starting point and $\Pi B(Y_p)$ is obtained by enumerating all elements of it independently of $\Pi B(Y_{p-1})$. If $\boldsymbol{w}^{p-1} \notin Y_p$, then a starting point can be obtained by solving

$$\underset{\boldsymbol{y} \in Y_p}{\text{maximize}}\ \boldsymbol{d}^{p\mathrm{T}} \boldsymbol{y} \tag{25}$$

This linear programming problem can be solved by a post-optimality technique.

4.4 An Example of Minimax Solution Procedure

Before the numerical experiment for the comparison among solution methods described above, we exemplify how the relaxation procedure algorithm (main routine) behaves. Note that the relaxation procedure is the same in all solution methods described above since the difference is only in the procedure for solving a subproblem.

Let us consider the following linear programming problem with uncertain objective coefficients:

$$\begin{aligned} \text{maximize} \quad & \gamma_1 x_1 + \gamma_2 x_2 \\ \text{subject to} \quad & 3x_1 + 4x_2 \le 42, \\ & 3x_1 + x_2 \le 24, \\ & x_1 \ge 0,\ 0 \le x_2 \le 9, \end{aligned} \tag{26}$$

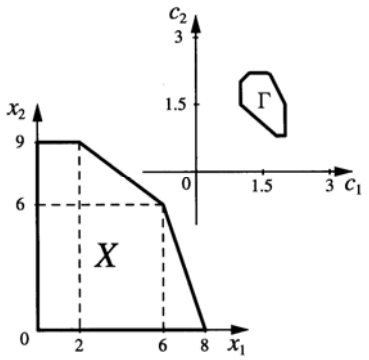


Figure 1: Feasible region X and Γ of Problem (26)

where the possible range of γ_1, γ_2, Γ is assumed to be given by

$$\begin{aligned}\Gamma = \{c = (c_1, c_2) \mid\ & 3.5 \le 2c_1 + c_2 \le 5.5, \\ & 3.4 \le c_1 + 2c_2 \le 6, \\ & -1 \le c_1 - c_2 \le 1.3, \\ & 1 \le c_1 \le 2, 0.8 \le c_2 \le 2.2\}\end{aligned} \tag{27}$$

The feasible region X and Γ are illustrated in Figure 1. Applying the relaxation procedure algorithm with setting ε =0.00001, we obtain a minimax regret solution as $(x_1, x_2)^{\mathrm{T}} = (4.94737, 6.78947)^{\mathrm{T}}$. The maximum regret is $R\left((4.94737, 6.78947)^{\mathrm{T}}\right) = 1.47368$. The process of the relaxation procedure is shown in Table 1.

5. A Numerical Experiment and the Result

5.1 The Numerical Experiment

In order to compare the computational efficiency of solution algorithms for the minimax regret solution described above, we did a numerical experiment by using a personal computer (Pentium III 700MHz, 128MB). Each solution algorithm is encoded by C language (GNU project C and C++ Compiler ver.2.95.2) on Free BSD ver.4.2. By the numerical experiment, we compare the solution algorithms by the computation time (CPU time, sec.). We consider many combinations of the number of decision variables, the number of constraints of X and the number of constraints of Γ, i.e., (n,m,p). To each combination, we generate 10 problems using random numbers.

Each problem is generated as follows. Using tangent hyperplanes of a randomly generated ellipsoid, we generate $X \subseteq \mathbf{R}^n_+$ (see Figure 2). We generate a polytope $\Theta \subseteq \mathbf{R}^n$ in terms of tangent hyperplanes of a randomly generated ellipsoid (see Figure 3(a)). Γ is defined as a polytope obtained by a parallel displacement of Θ so as to be inscribed in a cone defined by

$$\sum_{\substack{i=1 \\ i \ne j}}^{n} c_i - \delta c_j \le 0,\ j = 1,2,\ldots,n, \tag{28}$$

where $\delta > n-1$ is a predetermined value (see Figure 3(b)). The variety of directions $\boldsymbol{c}/|\boldsymbol{c}|$ in Γ increases as δ ineases. Namely, the generated problem with large δ will be more difficult than the one with small since the number of possibly optimal solutions increases as the variety of directions of objective coefficient vectors increases. In our experiment, e select three values for δ, i.e., $\delta = 4(n-1)$, $\delta = 2(n-1)$ and $\delta = 4(n-1)/3$.

Table 1: An Illustration of Relaxation Procedure Algorithm.

Step 1.	Select $\boldsymbol{c}^0 = (1,2)^{\mathrm{T}}$. We obtain $\boldsymbol{z}^0 = (2,9)^{\mathrm{T}} \in S(\boldsymbol{c}^0)$.
Step 2.	Set $r^0 = 0$, $k = 1$ and $\boldsymbol{x}^0 = \boldsymbol{z}^0$.
Step 3.	Solve Problem (8). We obtain $r^1 = 5.6$, $\boldsymbol{c}^1 = (2, 0.8)^{\mathrm{T}}$ and $\boldsymbol{z}^1 = (6,6)^{\mathrm{T}}$.
Step 4.	$r^1 - r^0 = 5.6 > 0.00001$. Continue.
Step 5.	Solve LP problem (9) with $k = 1$. Update $\boldsymbol{x}^0 = (4.94737, 6.78947)^{\mathrm{T}}$, $r^0 = 1.47368$ and $k = 2$. Return to Step 3.
Step 3.	Solve Problem (8). We obtain $r^2 = 1.47368$, $\boldsymbol{c}^2 = (2, 0.8)^{\mathrm{T}}$ and $\boldsymbol{z}^2 = (6,6)^{\mathrm{T}}$.
Step 4.	$r^2 - r^0 = 0 \le 0.00001$. Terminate with the solution $\boldsymbol{x} = (4.94737, 6.78947)^{\mathrm{T}}$.

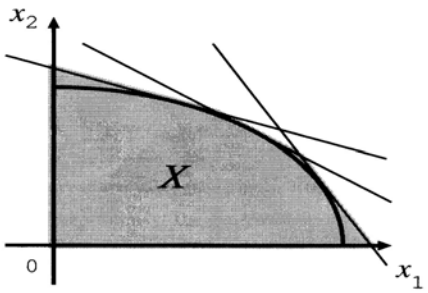
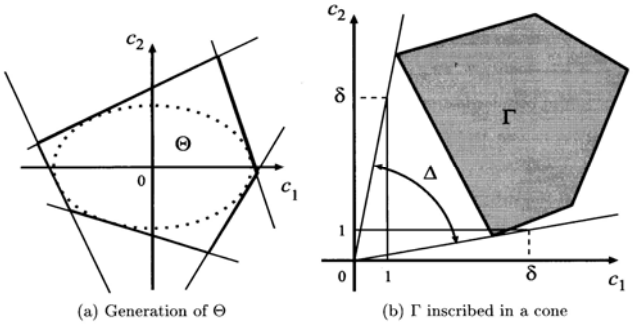


Figure 2: Generation of Constraints

5.2 Modification of the Relaxation Procedure

To some small-sized problems we applied the solution algorithms described in previous sections as a preliminary experiment. We observed that a lot of computation time (more than 1200 sec.) were necessary in the cutting hyperplane approach and that the computation time in cutting hyperplane approach was too large to compare with the others. From this fact, we may conclude that the cutting hyperplane approach is out of question and should be dismissed. However, this big computation time is caused by the fact that the cutting hyperplane method requires plenty of time for the convergence to an exact optimal solution though it converges relatively rapidly to a sub-optimal solution. To compute a minimax regret solution by the relaxation procedure, we should solve maximum regret problems (sub-problems) multiple times and thus, obtaining exact optimal solutions to the subproblems by the cutting hyperplane method costs a lot of computation time.



(a) Generation of Θ (b) Γ inscribed in a cone

Figure 3. Generation of Γ

Taking into account the rapid convergence to a sub-optimal solution in the cutting hyperplane method, we modified the relaxation procedure so that we can compare the computation time of the cutting hyperplane approach with the others. The main aim of Steps 3 and 4 of the relaxation procedure algorithm in Subsection 3.1 is just to check the existence of $\left(\boldsymbol{c}^k, \boldsymbol{z}^k\right)$ such that $r^k = \boldsymbol{c}^{k\mathrm{T}}\left(\boldsymbol{z}^k - \boldsymbol{x}^0\right) > r^0 + \varepsilon$. From this point of view, we may replace Steps 3 and 4 of the relaxation procedure algorithm with the following single step:

Step 3&4 Find $\left(\mathbf{c}^k, \mathbf{z}^k\right)$ such that
$r^k = \left(\boldsymbol{c}^k\right)^{\mathrm{T}} \boldsymbol{z}^k - \left(\boldsymbol{c}^k\right)^{\mathrm{T}} \boldsymbol{x}^0 > r^0 + \varepsilon, \boldsymbol{c}^k \in \Gamma$
and $\boldsymbol{z}^k \in X$. If there exists no such solution, then terminate the algorithm. In this case, the obtained minimax regret solution is $\mathbf{x}^0$. Otherwise, proceed to Step 5.

Whereas the original Steps 3 and 4 are trying to find the deepest cut of the relaxed constraints of Problem (9), the above single step is trying to find just a cut. By this modification, we reduce the computation time for Steps 3 and 4, but may increase the number of iterations of the relaxation procedure. Thus the improvement by this modification depends on how the solution method for sub-problem (8) can find a good sub-optimal solution in early iterations. Moreover, this modification can be effective only for the cases when a lot of computation time is consumed for solving sub-problem (8). Once all elements of $\prod B$ are enumerated, solving sub-problem(8) does not take a lot of computation time. Hence, this modification is not effective for two-phase approach.
We equipped the modification in all approaches except two-phase approach. As the result, we compare eight approaches listed in Table 2.

5.3 Experimental Results

We solved all generated problems by the eight approaches and measured each computation time. The obtained average computation time (CPU time, sec.) for each approach is listed in Tables 3 and 4. In the experiment, the computation time for solving a problem is limited to 1200 (sec.). The computation time of problems over 1200 (sec.) is treat as 1200 (sec.) in the calculation of the averages. Instead, we put a sign `+' and the number of such problems in parentheses after the averages.

Table 2. Eight Approaches Compared in The Experiment

approach	result
two-phase approach without modification	'2phase' in Table 3
bilevel programming approach without modification	'bilevel' in Table 3
outer approximation approach without modification	'outer' in Table 3
combination of the outer approximation and cutting hyperplane approaches without modification	'combi.' in Table 3
bilevel programming approach with modification	'bilevel' in Table 4
outer approximation approach with modification	'outer' in Table 4
cutting hyperplane approach with modification	'cutting' in Table 4
combination of the outer approximation and cutting hyperplane approaches with modification	'combi.' in Table 4

In Tables 3 and 4, we can also observe that the computation time decreases as δ decreases because the larger δ is, the more difficult the problem is.

From Table 3, in the case when we do not introduce the modification, we can observe the following. When the problem size is not large, the two phase approach is as efficient as the outer approximation method, but when the problem size is large, the outer approximation approach seems to be the most efficient. However, the efficiency of the outer approximation approach relative to two phase approach is not very drastic. The combination of the outer approximation and cutting hyperplane methods is worse than the outer approximation method. This is because we cannot utilize the last vertex set obtained by solving the previous maximum regret problem as the initial vertex set for the current maximum regret problem in the mixed methods while we can utilize it in the outer approximation method. The bilevel programming approach is not very efficient.

Comparing Tables 3 and 4, we observe that the computation time is improved by the modification in the outer approximation approach and in the combination of the outer approximation and cutting hyperplane approaches. There is no improvement in the bilevel programming approach. From these results, we could say that a good sub-optimal solution to the subproblem can be obtained in early iterations of algorithms based on the outer approximation method and the combination of the outer approximation and cutting hyperplane methods. In the bilevel programming approach, a good sub-optimal solution does not seem to be obtained in early iterations. By the modification, the cutting hyperplane approach is improved very much, so that the

Table 3. The Average Computation Time of Each Approach Without Modification

$\delta = 4(n-1)$									
(n,m,p)	2phase	bilevel	outer	combi.	(n,m,p)	2phase	bilevel	outer	combi.
(5, 10, 10)	0.049	0.369	0.031	0.053	(10, 20, 20)	21.590	68.043	14.996	20.419
(5, 10, 15)	0.105	0.762	0.091	0.120	(10, 20, 30)	65.684	147.849	32.177	65.554
(5, 10, 20)	0.254	1.443	0.199	0.296	(10, 20, 40)	144.177	285.308	56.503	113.727
(5, 15, 10)	0.092	1.418	0.070	0.133	(10, 30, 20)	36.948	167.365	21.296	27.056
(5, 15, 15)	0.133	1.936	0.106	0.156	(10, 30, 30)	138.809	678+(3)	65.999	130.476
(5, 15, 20)	0.229	3.235	0.154	0.219	(10, 30, 40)	174.652	985+(2)	79.527	151.215
(5, 20, 10)	0.104	1.959	0.064	0.083	(10, 40, 20)	171.663	662.288	43.938	70.659
(5, 20, 15)	0.294	5.465	0.183	0.320	(10, 40, 30)	239.486	981+(5)	86.251	176.613
(5, 20, 20)	0.434	7.648	0.344	0.455	(10, 40, 40)	593.280	1164+(9)	226.213	289.854
$\delta = 2(n-1)$									
(n,m,p)	2phase	bilevel	outer	combi.	(n,m,p)	2phase	bilevel	outer	combi.
(5, 10, 10)	0.039	0.537	0.040	0.061	(10, 20, 20)	4.219	68.367	3.994	9.455
(5, 10, 15)	0.064	0.692	0.063	0.088	(10, 20, 30)	20.412	167.396	13.107	33.068
(5, 10, 20)	0.120	1.208	0.137	0.159	(10, 20, 40)	18.950	204.136	13.799	26.073
(5, 15, 10)	0.035	1.110	0.036	0.058	(10, 30, 20)	23.891	309.705	17.986	30.003
(5, 15, 15)	0.090	2.510	0.085	0.150	(10, 30, 30)	35.429	671+(1)	22.831	65.360
(5, 15, 20)	0.113	3.238	0.106	0.161	(10, 30, 40)	142.160	871+(4)	91.763	161.457
(5, 20, 10)	0.069	2.583	0.042	0.074	(10, 40, 20)	45.734	777+(5)	32.706	75.474
(5, 20, 15)	0.135	5.160	0.114	0.206	(10, 40, 30)	57.780	1051+(6)	31.483	84.899
(5, 20, 20)	0.258	8.594	0.240	0.355	(10, 40, 40)	150.838	1200+(10)	90.451	164.106
$\delta = 4(n-1)/3$									
(n,m,p)	2phase	bilevel	outer	combi.	(n,m,p)	2phase	bilevel	outer	combi.
(5, 10, 10)	0.016	0.445	0.022	0.032	(10, 20, 20)	2.026	58.401	2.124	6.176
(5, 10, 15)	0.024	0.799	0.029	0.058	(10, 20, 30)	2.263	109.744	3.044	8.113
(5, 10, 20)	0.041	1.135	0.048	0.082	(10, 20, 40)	9.214	245.334	9.278	16.350
(5, 15, 10)	0.022	0.972	0.021	0.056	(10, 30, 20)	3.210	260.199	3.182	7.575
(5, 15, 15)	0.039	2.192	0.042	0.095	(10, 30, 30)	7.989	661+(2)	9.385	23.593
(5, 15, 20)	0.079	3.800	0.092	0.174	(10, 30, 40)	6.219	643+(1)	6.675	19.436
(5, 20, 10)	0.033	3.389	0.035	0.087	(10, 40, 20)	6.270	708+(2)	5.720	16.977
(5, 20, 15)	0.038	4.126	0.040	0.095	(10, 40, 30)	10.722	1120+(7)	11.496	33.562
(5, 20, 20)	0.083	5.766	0.087	0.159	(10, 40, 40)	49.415	1156+(9)	39.277	84.762

Table 4. The Average Computation Time of Each Approach With Modification

$\delta = 4(n-1)$									
(n,m,p)	bilevel	outer	cutting	combi.	(n,m,p)	bilevel	outer	cutting	combi.
(5, 10, 10)	0.635	0.018	0.081	0.033	(10, 20, 20)	89.904	9.289	121+(1)	5.842
(5, 10, 15)	1.008	0.046	1.276	0.060	(10, 20, 30)	157.579	15.778	146.475	20.425
(5, 10, 20)	2.026	0.100	5.120	0.124	(10, 20, 40)	366.403	45.984	141+(1)	45.920
(5, 15, 10)	1.951	0.040	4.573	0.056	(10, 30, 20)	250.119	11.303	223+(1)	4.984
(5, 15, 15)	2.570	0.061	1.155	0.069	(10, 30, 30)	645+(1)	45.766	603+(5)	25.724
(5, 15, 20)	3.545	0.116	7.866	0.113	(10, 30, 40)	1108+(4)	40.767	632+(5)	40.380
(5, 20, 10)	2.345	0.041	9.314	0.059	(10, 40, 20)	895+(3)	25.611	244+(2)	14.899
(5, 20, 15)	6.493	0.086	43.771	0.120	(10, 40, 30)	1138+(7)	62.626	139+(1)	37.527
(5, 20, 20)	10.106	0.207	4.873	0.202	(10, 40, 40)	1200+(10)	156.746	365+(3)	86.464
$\delta = 2(n-1)$									
(n,m,p)	bilevel	outer	cutting	combi.	(n,m,p)	bilevel	outer	cutting	combi.
(5, 10, 10)	0.825	0.023	0.180	0.043	(10, 20, 20)	76.264	1.970	483+(4)	3.642
(5, 10, 15)	0.934	0.053	0.320	0.053	(10, 20, 30)	195.191	10.528	259+(2)	9.139
(5, 10, 20)	1.468	0.092	0.901	0.094	(10, 20, 40)	182.379	9.192	349+(2)	11.117
(5, 15, 10)	1.318	0.025	0.547	0.034	(10, 30, 20)	474.758	11.713	390+(3)	13.521
(5, 15, 15)	3.136	0.041	0.734	0.071	(10, 30, 30)	629+(2)	13.502	614+(4)	22.004
(5, 15, 20)	3.298	0.069	3.186	0.108	(10, 30, 40)	801+(3)	62.597	398+(3)	57.604
(5, 20, 10)	3.143	0.027	0.946	0.056	(10, 40, 20)	831+(6)	19.632	243+(2)	20.708
(5, 20, 15)	5.530	0.081	14.392	0.117	(10, 40, 30)	985+(6)	23.365	245+(2)	18.354
(5, 20, 20)	8.446	0.170	7.808	0.148	(10, 40, 40)	1035+(8)	54.703	505+(4)	45.880
$\delta = 4(n-1)/3$									
(n,m,p)	bilevel	outer	cutting	combi.	(n,m,p)	bilevel	outer	cutting	combi.
(5, 10, 10)	0.481	0.014	0.639	0.020	(10, 20, 20)	52.171	1.105	241+(2)	3.421
(5, 10, 15)	0.575	0.022	0.075	0.035	(10, 20, 30)	89.564	1.968	138+(1)	3.737
(5, 10, 20)	1.138	0.036	0.118	0.042	(10, 20, 40)	157.958	5.372	422+(3)	7.164
(5, 15, 10)	0.880	0.014	0.076	0.026	(10, 30, 20)	288.539	1.825	121+(1)	3.324
(5, 15, 15)	1.492	0.034	0.637	0.051	(10, 30, 30)	421.900	6.112	122+(1)	10.444
(5, 15, 20)	2.807	0.074	1.231	0.074	(10, 30, 40)	403.471	5.006	482+(3)	6.490
(5, 20, 10)	2.840	0.031	0.187	0.047	(10, 40, 20)	582+(2)	3.615	278+(2)	5.458
(5, 20, 15)	2.379	0.032	0.186	0.061	(10, 40, 30)	807+(2)	8.614	489+(4)	12.575
(5, 20, 20)	3.944	0.071	4.352	0.097	(10, 40, 40)	1022+(6)	27.873	244+(2)	36.345

computation time can be measured to be compared with the other approaches. Nevertheless, it is by far inferior to the outer approximation approach and the combination of the outer approximation and cutting hyperplane approaches.

From Tables 3 and 4, we can conclude that the outer approximation approach with modification and the combination of the outer approximation and cutting hyperplane approaches with modification are most efficient in the sense of the average computation time.

6. Concluding Remarks

In this paper, we discuss solution algorithms for minimax regret solutions to linear programming problems whose objective function coefficients are unknown but known in given polytopes. An outer approximation method and a cutting hyperplane method are designed to solve minimax regret problems. Moreover, a combination of the outer approximation and cutting hyperplane approaches is considered. Since the cutting hyperplane approach takes a lot of computation time, we introduced a simple modification of the relaxation procedure.

Four approaches without modification, i.e., a two-phase approach, a bilevel programming approach, an outer approximation approach and a combination of the outer approximation and cutting hyperplane approaches, and four approaches with modification, i.e., a bilevel programming approach, an outer approximation approach, a cutting hyperplane approach and a combination of the outer approximation and cutting hyperplane approaches, were compared in computational efficiency by a numerical experiment. The results showed that the outer approximation approach with modification and the combination of the outer approximation and cutting hyperplane approaches with modification were the most computationally efficient.

Introduction of other methods such as inner approximation and reformulation techniques developed for bilinear programming problems will be studied to develop new solution algorithms for minimax regret problems in the future.

Acknowledgements

This research was partly supported by the Ministry of Education, Science, Sports and Culture, Grant-in-Aid Scientific Research for Encouragement of Young Scientists, No.09780412. The authors also wish to thank Hidetaka Higashitani, Koichi Matsumoto and Michiyoshi Nishimura for their help in numerical experiments.

Reference

[1] M. Inuiguchi and M. Sakawa, ”Possible and Necessary Optimality Tests in Possibilistic Linear Programming Problems,” *Fuzzy Sets and Systems*, Vol. 67, pp.29-46,1994.

[2] M. Inuiguchi and M. Sakawa, “Minimax Regret Solutions to Linear Programming Problems with an Interval Objective Function,” *European Journal of Operational Research*, Vol.86, pp.526-536, 1995.

[3] M. Inuiguchi and M. Sakawa, “Minimax Regret

Solution Algorithms for Linear Program with Convex Polyhedral Objective Coefficients," *Proc. of 2nd European Workshop on Fuzzy Decision Analysis and Neural Networks for Management, Planning and Optimization*, Dortmund, Germany, June, 1997, pp.116-125.

[4] H. E. Mausser and M. Laguna, "A New Mixed Integer Formulation for the Maximum Regret Problem," *International Transactions in Operational Research*, Vol. 5, pp.389-403, 1998.

[5] M. Inuiguchi and Y. Kume, "Minimax Regret in Linear Programming Problems with an Interval Objective Function," in: G. H. Tzeng, H. F. Wang, U. P. Wen and P. L. Yu (eds.), *Multiple Criteria Decision Making*, Springer-Verlag, New York, NY, pp.65-74, 1994.

[6] K. Shimizu and E. Aiyoshi, "Necessary Conditions for Min-max Problems and Algorithms by A Relaxation Procedure," *IEEE Trans. Automatic Control*, Vol. AC-25, No.1, pp.62-66, 1980.

[7] M. Inuiguchi, H. Higashitani and T. Tanino, "On Enumeration of Possibly Optimal Extreme Points in Linear Programming Problems with Interactive Possibilistic Variables," *Proc. of 7th European Congress on Intelligent Techniques and Soft Computing*: CD Rom, Aachen, Germany, CC5-3-12852_P.pdf (file name), 1999.

[8] R. Horst and H. Tuy, *Global Optimization*: *Deterministic Approaches*, *Third, Revised and Enlarged Edition*, Berlin: Springer-Verlag, 1995.

[9] H. D. Sherali and C. M. Shetty, "A Finitely Convergent Algorithm for Bilinear Programming Problems Using Polar Cuts and Disjunctive Face Cut," *Mathematical Programming*, Vol. 19, pp. 14-31, 1980.

Mashairo Inuiguchi is an associate professor at the Department of Electronics and Information Systems, Graduate School of Engineering, Osaka University.